

WHAT IS CLAIMED IS:

1 1. A method used in a concurrent program analysis for detecting potential race
2 conditions, such as data races, in a computer program, comprising:
3 receiving a source code of the computer program, the source code including an
4 element annotated as either thread-local or thread-shared;
5 determining if the element is annotated as thread-shared or thread-local; and
6 verifying the validity of the thread-local annotation if the element is annotated as
7 thread-local,
8 wherein an invalid thread-local annotation may cause a race condition.

1 2. The method of claim 1, wherein the computer program can spawn a plurality of
2 threads that are capable of being executed concurrently, the method further comprising:
3 indicating a race condition warning or error if upon verifying the validity of the
4 thread-local annotation of the element it is determined that the element is, in
5 fact, visible from more than one, rather than one and only one, of the plurality
6 of threads.

1 3. The method of claim 1 wherein, for any instance in which it is determined that the
2 element is annotated as thread-shared, the method further comprises:
3 verifying that the element does not include a portion annotated as thread-local and/or
4 a link to another element that is annotated as thread-local; and
5 indicating a race condition warning or error if the portion and/or the other element are
6 annotated as thread-local.

1 4. The method of claim 1, wherein the element can be a global addressable resource and,
2 if so, the method further comprises:

3 verifying that the element does not include a portion annotated as thread-local and/or
4 a link to another element that is annotated as thread-local; and
5 indicating a race condition warning or error if the portion and/or the other element are
6 annotated as thread-local.

1 5. The method of claim 3, wherein the element is a class structure, an object, a data
2 structure or a record, the portion of which respectively being a class object, an
3 attribute, a structure element, or a field.

1 6. The method of claim 1 wherein, for any instance in which it is determined that the
2 element is annotated as thread-shared and includes a pointer or a reference to a different
3 element, the method further comprises:
4 verifying that the different element is not annotated as thread-local; and
5 indicating a race condition warning or error if the different element is annotated as
6 thread-local.

1 7. The method of claim 1, further comprising:
2 indicating a race condition warning or error if the element is thread-shared annotated
3 and it is determined that the at least one portion of the element points to
4 another element of the source code that is thread-local.

1 8. The method of claim 1, wherein the computer program can spawn a plurality of
2 threads that are capable of being executed concurrently, and wherein verifying the validity of
3 the thread-local annotation includes
4 checking whether at least one portion of the element, or another element
5 pointed to by the element, is visible from more than one, rather than one and only one,
6 of the plurality of threads, and
7 checking whether upon creation of a new thread of the plurality of threads the
8 element is passed to the new thread,

9 wherein a race condition warning or error is indicated if the element and/or the
10 other element are annotated as thread-local but are visible from more than one, rather
11 than one and only one, of the plurality of threads.

1 9. The method of claim 1, wherein the computer program can spawn a plurality of
2 threads that are capable of being executed concurrently, and wherein verifying the validity of
3 the thread-local annotation includes

4 checking, if the element is annotated as thread-shared, whether each
5 portion of the element is also annotated as thread-shared,
6 checking, if the element is visible from more than one of the plurality of
7 threads, whether the element is annotated as thread-shared, and
8 checking, if the element is passed into a new thread that is spawned from one
9 of the plurality of threads, whether the element is annotated as
10 thread-shared,
11 wherein an invalid thread-local annotation can prompt a warning indication.

1 10. The method of claim 1, further comprising:
2 checking whether a sub-element is derived from the element and, if so,
3 checking, if the element is annotated as thread-local, whether the sub-element
4 is also annotated as thread-local,
5 checking, if the element is annotated as thread-shared,
6 whether the sub-element is also annotated as thread-shared, or
7 whether the sub-element is annotated as thread-local, and the sub-
8 element does not override methods declared in the element and
9 the element is not typecast to the sub-element.

1 11. The method of claim 10 wherein, for any instance in which it is determined that the
2 sub-element is derived from the element, the method further comprises:
3 providing a race condition warning or error indication

4 if the element is annotated as thread-local and the sub-element is not
5 annotated as thread-local, or
6 if the element is annotated as thread-shared, the sub-element is
7 annotated as thread-local, and either
8 the sub-element overrides methods declared in the element, or
9 the element is typecast to the sub-element.

1 12. An apparatus for concurrent program analysis, comprising:
2 means for receiving source code of a computer program, the source code including
3 an element annotated as either thread-local or thread-shared;
4 means for type checking the source code; and
5 means for checking annotations located either inside or in series with the type
6 checking means, including
7 means for determining whether the element is annotated as thread-shared or
8 thread-local; and
9 means for verifying the validity of the thread-local annotation if the element is
10 annotated as thread-local,
11 wherein an invalid thread-local annotation may cause a race condition such as a data
12 race.

1 13. The apparatus of claim 12, further comprising:
2 means for parsing the source code; and
3 means for creating from the source code an abstract syntax tree.

1 14. The apparatus of claim 12, wherein the computer program can spawn a plurality of
2 threads that are capable of being executed concurrently, and wherein the means for checking
3 annotations further includes
4 means for checking, if the element is annotated as thread-local, whether the
5 element is visible from more than one of the plurality of threads,

6 means for checking, if the element is annotated as thread-shared, whether each
7 portion of the element is also annotated as thread-shared,
8 and
9 means for checking, if the element is passed into a new thread that is spawned
10 from one of the plurality of threads, whether the element is annotated
11 as thread-local,
12 wherein an invalid thread-local annotation can prompt the apparatus to provide
13 a warning indication.

1 15. The apparatus of claim 12, wherein the means for checking annotations further
2 includes
3 means for checking whether a sub-element is derived from the element and, if
4 so,
5 means for checking, if the element is annotated as thread-local,
6 whether the sub-element is also annotated as thread-local,
7 means for checking, if the element is annotated as thread-shared,
8 whether the sub-element is also annotated as thread-shared, or
9 whether the sub-element is annotated as thread-local, and the
10 sub-element does not override methods declared in the
11 element and the element is not typecast to the sub-
12 element.

1 16. The method of claim 15, wherein, for any instance in which it is determined that the
2 sub-element is derived from the element, the means for checking annotations further includes
3 means for providing a race condition warning or error indication
4 if the element is annotated as thread-local and the sub-element is not
5 annotated as thread-local, or
6 if the element is annotated as thread-shared, the sub-element is
7 annotated as thread-local, and either

8 the sub-element overrides methods declared in the element, or
9 the element is typecast to the sub-element.

1 17. A system for concurrent program analysis having a computer readable medium
2 embodying program code for detecting potential race conditions, such as data races, in a
3 computer program, including instructions for causing the system to:
4 receive a source code of the computer program, the source code including an
5 element annotated as either thread-local or thread-shared;
6 determine if the element is annotated as thread-shared or thread-local; and
7 verify the validity of the thread-local annotation if the element is annotated as thread-
8 local, wherein an invalid thread-local annotation may cause a race condition.